UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

GRADO EN INGENIERÍA DEL SOFTWARE

PROYECTO FIN DE GRADO

Desarrollo de sistema para organizar eventos entre grupos.

Autor: Sergio Fernández Jiménez
Tutor: Francisco Serradilla García
Fecha: Marzo, 2015

Figuras

Capítulo 1

1 Introducción

1.1 Resumen del proyecto

Organizar actividades y proyectos entre varias personas o tomar decisiones conjuntas son cuestiones a las que se enfrenta cualquier individuo en su día a día. El simple hecho de coordinar o poner de acuerdo a un grupo reducido de personas puede llegar a suponer un gran problema ya que cada participante tiene sus propias preferencias y, en ocasiones, es difícil conseguir encajarlas con las demás del grupo.

Este proyecto, llamado "DealtDay", surge para facilitar esta labor.

La idea nace ante la necesidad de organizar, de forma fácil e intuitiva, a un grupo de personas para, por ejemplo, concretar una reunión, quedar para ir a dar una vuelta, decidir qué película ver, etc.

Este proyecto se ha desarrollado basándose en el sistema actual de relaciones con el que se han creado la mayoría de las redes sociales que hoy conocemos.

Como medio para poder hacer uso del proyecto se ha construido una aplicación web que, gracias a las decisiones de diseño tomadas, se puede usar tanto en un ordenador, una tablet o un smartphone. Este punto se considera fundamental ya que cada vez más personas están dejando de lado los ordenadores corrientes para dar paso al uso de las nuevas tecnologías.

Además, se ha creado una API REST, lo que nos permite utilizar todas las funcionalidades de la aplicación desde cualquier sistema que pueda realizar peticiones http.

En este proyecto en concreto se realizará la parte del desarrollo de la API, el cliente web y el despliegue de la aplicación en un servidor web para realizar las pruebas pertinentes.

1.2 Abstract

To organize activities and projects between several people or make joint decisions are issues to which any persons faces every day. The simple fact of coordinate or coming to an agreement with a group of person could be a major problem, since each participant has their own preferences and often fails when tries to fit them with the group.

This project, called “DealtDay”, is born to facilitate this task.

The idea arises of how to achieve organize a group of people in a easily and intuitively way in order to arrange a meeting, be able to go for a walk, decide what movie to see or simply vote a choice between a users group.

This project has been developed based on the current relation system that has been created in the most social networks we know.

As a means of making use of the project a web application has been built, that thanks to the design decisions taken it can be used in a computer, tablet or smartphone. This is an essential point because more and more people are abandoning the current computers to make way for the use of new technologies.
Also, a REST API has been created, which allows us to use all the features of the application from any system able to make http requests.

In this particular project , I have done the development of the API, web client and the application deployment on a web server in order to test it.

1.3 Objetivos

El principal objetivo del proyecto es desarrollar una aplicación que permita tomar decisiones de forma fácil entre un grupo de personas o recoger opiniones entre varios usuarios sobre un asunto que pueda resultar de relevancia.

Para lograr este propósito, se desarrollará una API que permita realizar la gestión de esta toma de decisiones; una vez concluida ésta, se llevará a cabo la implementación de un sistema de amigos para poder gestionarlos e invitarlos a participar en la toma de decisiones.

El siguiente paso será elaborar una aplicación web que haga uso de la API desarrollada, lo que servirá también para obtener "feedback" para mejorar.

Por último, se desplegará todo el sistema desarrollado en un entorno de producción para poder realizar tests con usuarios reales.

Capítulo 2

2 Tecnologías utilizadas

En este proyecto se ha llevado a cabo una innovación dentro de las típicas tecnologías de desarrollo web, como pueden ser jQuery y Bootstrap, combinadas con algunas ya cada vez más clásicas para la parte de "backend" como son Django y MySQL.

En el entorno de la informática, es importante estar continuamente actualizándose y aprendiendo tecnologías nuevas. Cada vez es mayor la información que nos rodea y las formas de trabajar están en continua evolución; aunque conocer todas las diferentes maneras de desarrollo web es prácticamente imposible, un alto conocimiento y dominio de ellas facilitará, seguro, nuestro desarrollo profesional en el futuro.

Las tecnologías utilizadas en la parte del desarrollo son tecnologías web básicas como HTML5, CSS3 y Javascript.

A continuación, se va a enumerar los diferentes frameworks que se han utilizado para después pasar a describirlos.

- AngularJS
- Angular-Material
- SASS
- Grunt
- Git
- Django
- Django-Rest-Framework
- MySQL
- Nginx

2.1 AngularJS

Éste es el framework que se eligió para sustituir a jQuery. Está basado en Javascript, un lenguaje interpretado, que corre en el lado del cliente (navegador) y trata de poner orden al caos en el que puede llegar a convertirse un proyecto desarrollado mediante Javacript o jQuery.

AngularJS aplica el patrón de diseño MVC (Modelo Vista Controlador) y marca una forma de trabajar clara, aunque también permite libertad al programador, como poder manipular el "DOM", ya que incluye una versión "lite" de jQuery, llamada "jQLite".

Con este framework las aplicaciones que desarrollaremos se denominan "Single View Application", esto quiere decir que, cuando estamos navegando a través de las diferentes secciones de la página web, ésta no se recarga entera si no que se va actualizando dinámicamente según los datos que vayamos pidiendo.

AngularJS es una tecnología relativamente nueva ya que su versión 1.0 fue lanzada en junio de 2012 y actualmente se encuentran en la versión 1.3 .

A continuación, podemos ver un cuadro comparativo donde podemos observar el aumento de popularidad que AngularJS ha sufrido frente a dos de sus actuales competidores, BackboneJS y EmberJS. Los datos son de 2012 a 2015.

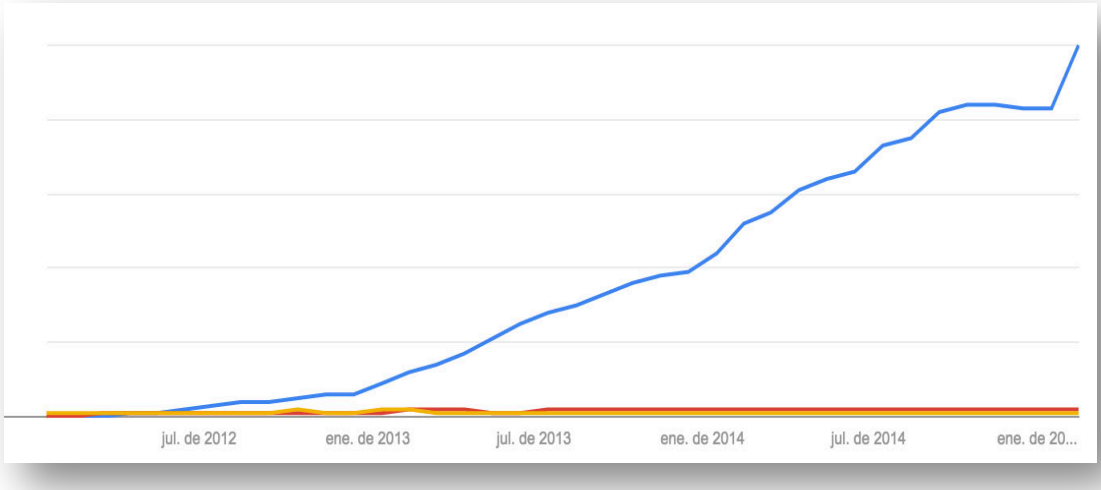


Figura 1: Gráfico de comparativa de popularidad.

Respecto a su curva de aprendizaje de este "framework", he de decir que lo fácil lo hace muy fácil, y lo difícil, en algunas ocasiones, muy difícil. Su curva de aprendizaje es muy irregular, con constantes zonas de subidas y bajadas hasta llegar a la cumbre.

Sin embargo, en mi opinión, creo que el esfuerzo merece la pena ya que facilita la programación en muchos aspectos.

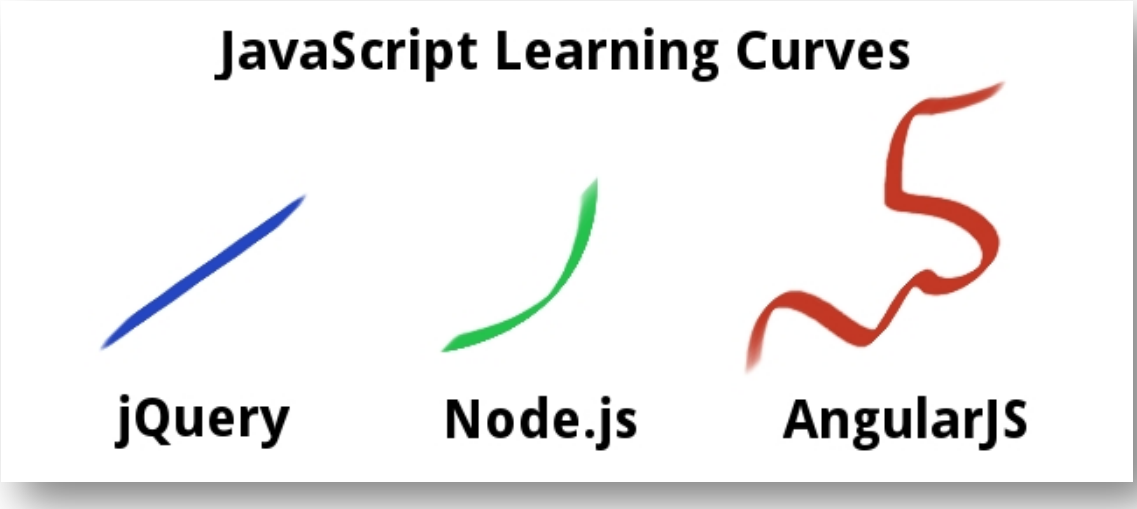


Figura 2: Comparación de curvas de aprendizaje jQuery, Node.js y AngularJS.

2.2 Angular-Material

Este es el framework que se decidió utilizar para desarrollar el aspecto visual de la página web. Está basado en los principios de diseño de "Material Design" de Google y ha sido adaptado para poder usarlo con AngularJS.

Se trata de una especificación para crear un sistema visual, interactivo y uniforme en diferentes resoluciones y plataformas. Esto nos permite construir una sola aplicación web que nos sirve tanto para la versión de escritorio como para la versión móvil.

Uno de los pilares fundamentales de este "framework" es el "diseño adaptativo" que tiene como objetivo adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla. Hoy en día las páginas web se visualizan en multitud de tipos de dispositivos ("tablets", "smartphones", ordenadores…) y multitud de tamaños de pantalla. Con el diseño web adaptativo se pretende tener una visualización adecuada a cada dispositivo con un único diseño.

El esfuerzo de implementar este sistema en el proyecto fue grande ya que cuando se empezó el desarrollo se encontraba en versión 0.5 ("alpha") y durante la realización del mismo ha ido evolucionando bastante, encontrándose actualmente en su versión 0.8, cada vez más cerca de la primera versión "beta" 1.0.

A continuación se adjuntan dos imágenes de cómo se vería la aplicación web en su versión de escritorio y en un dispositivo móvil.

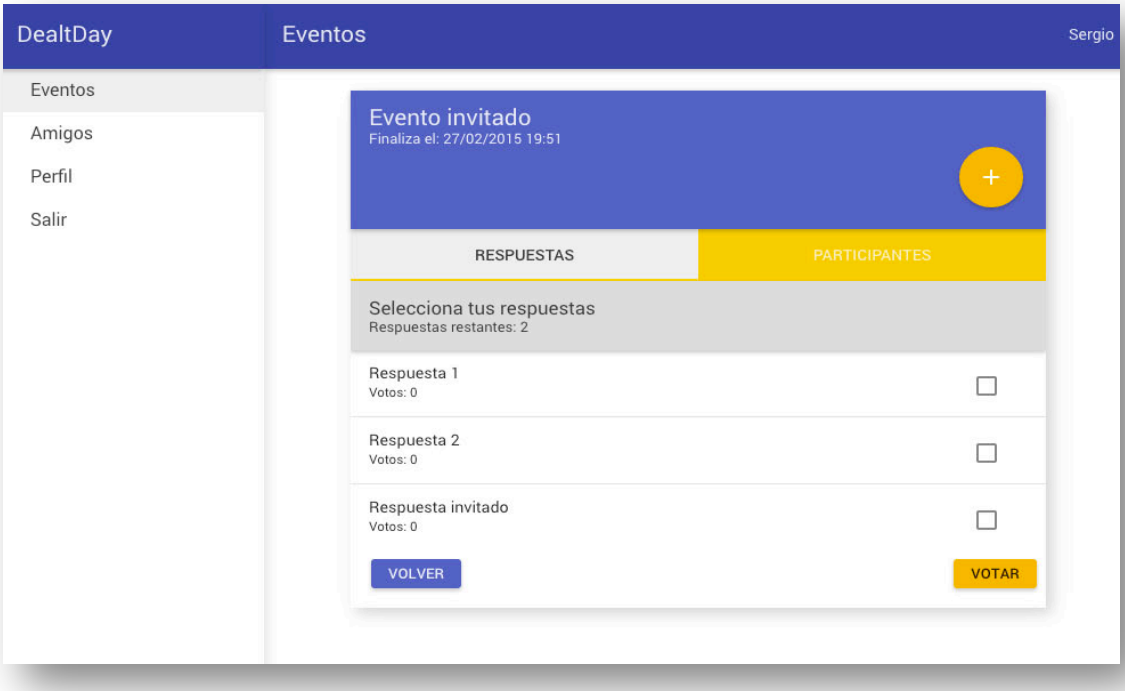


Figura 3: Ejemplo de pantalla a resolución mayor de 960 píxeles.

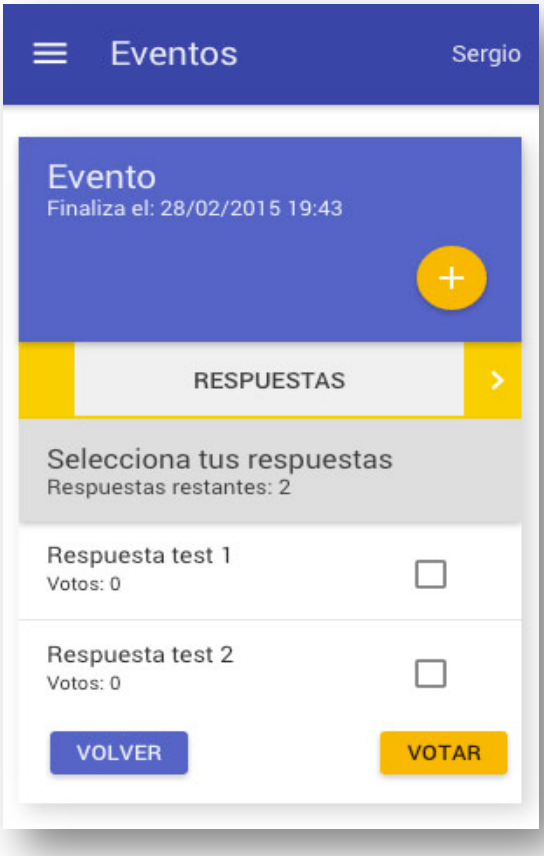


Figura 4: Ejemplo de pantalla a resolución menor de 960 píxeles.

2.3 SASS/SCSS

SASS ("*Syntactically Awesome StyleSheets*") es una extensión del lenguaje CSS que añade características muy potentes al mismo.

Algunas de las características más destacadas de SASS es que nos permite definir variables, reglas de anidamiento, "mixins", algunas directivas de control, etc.

SASS trabaja con dos tipos de archivos ".sass", perteneciente a SASS, y ".scss", perteneciente a SCSS ("Sassy CSS").

Se trata de un preprocesador de CSS que nos generará uno a varios archivos ".css" formateados y listos para incluir en nuestro código, ya que los archivos "sass" o "scss" no son los que se incluyen en el proyecto.

2.4 Grunt

Se trata de una librería Javascript que nos permite configurar tareas para que se ejecuten de forma automática.

Grunt funciona a través de pequeños paquetes que instalamos dependiendo de las funcionalidades que queramos automatizar. Este framework nos proporciona un gran ahorro de tiempo cuando nos encontramos desarrollando o desplegando una aplicación web ya que nos permite despreocuparnos de aquellas tareas que son repetitivas.

El único requisito para usar Grunt es tener instalado "Node.js" y su sistema de gestión de paquetes "npm", a través del cual instalaremos los paquetes que queramos.

Los plugins que se han utilizado en este proyecto son los siguientes:

- **"Grunt-autoprefixer":** este plugin permite ahorrarse la repetitiva tarea de tener que definir las reglas CSS para todos los navegadores a través de sus prefijos específicos (-webkit, -moz). Con este módulo únicamente debemos definir la regla general y es él quien genera las específicas para cada navegador.
- **"Grunt-concurrent":** permite correr tareas grunt de manera concurrente.
- **"Grunt-contrib-concat":** a través de este plugin concatenamos todos los archivos que queramos uno detrás del otro lo que nos permite, por ejemplo, generar todos los ficheros Javascript comprimidos en uno sólo y así realizar, únicamente, una petición al servidor.

- **“Grunt-contrib-sass”:** con este módulo compilamos los archivos .sass a .css .
- **“Grunt-contrib-watch”:** permite ejecutar tareas definidas sobre ficheros que están siendo observados, sobre los que se produce una modificación o son eliminados.

Estos son algunos de los que se han utilizado en el desarrollo del proyecto aunque hay muchos más; actualmente, el sistema consta de 4285 módulos y podemos encontrar uno para casi cualquier tarea que queramos realizar.

2.5 Git

Git es un software de control de versiones que permite realizar un seguimiento de los cambios en archivos y, si es necesario, restaurar las versiones anteriores.

Junto a Git se ha utilizado Git-Flow, una forma de trabajar en Git que permite facilitar la gestión de ramas y el flujo de trabajo.

El trabajo con Git-Flow se desarrolla en dos ramas principales, la rama “master” y la rama “develop”. Todos los “commits” de la rama “master” deben de estar preparados para subir a producción, y todos los de la rama “develop” conformarán la siguiente versión del proyecto.

Además git-flow nos propone tres ramas auxiliares que son la rama “feature”, que nos permite ir desarrollando las diferentes funcionalidades para luego unirla a “develop”, la rama “release” que se utilizara para corregir los últimos arreglos antes de subir a producción, una vez finalizada se unirá a “develop” y “master”, y por último la rama “hotfix”, que se utilizará para corregir “bugs” de la versión de producción y se unirá a la rama “develop” y “master” también.

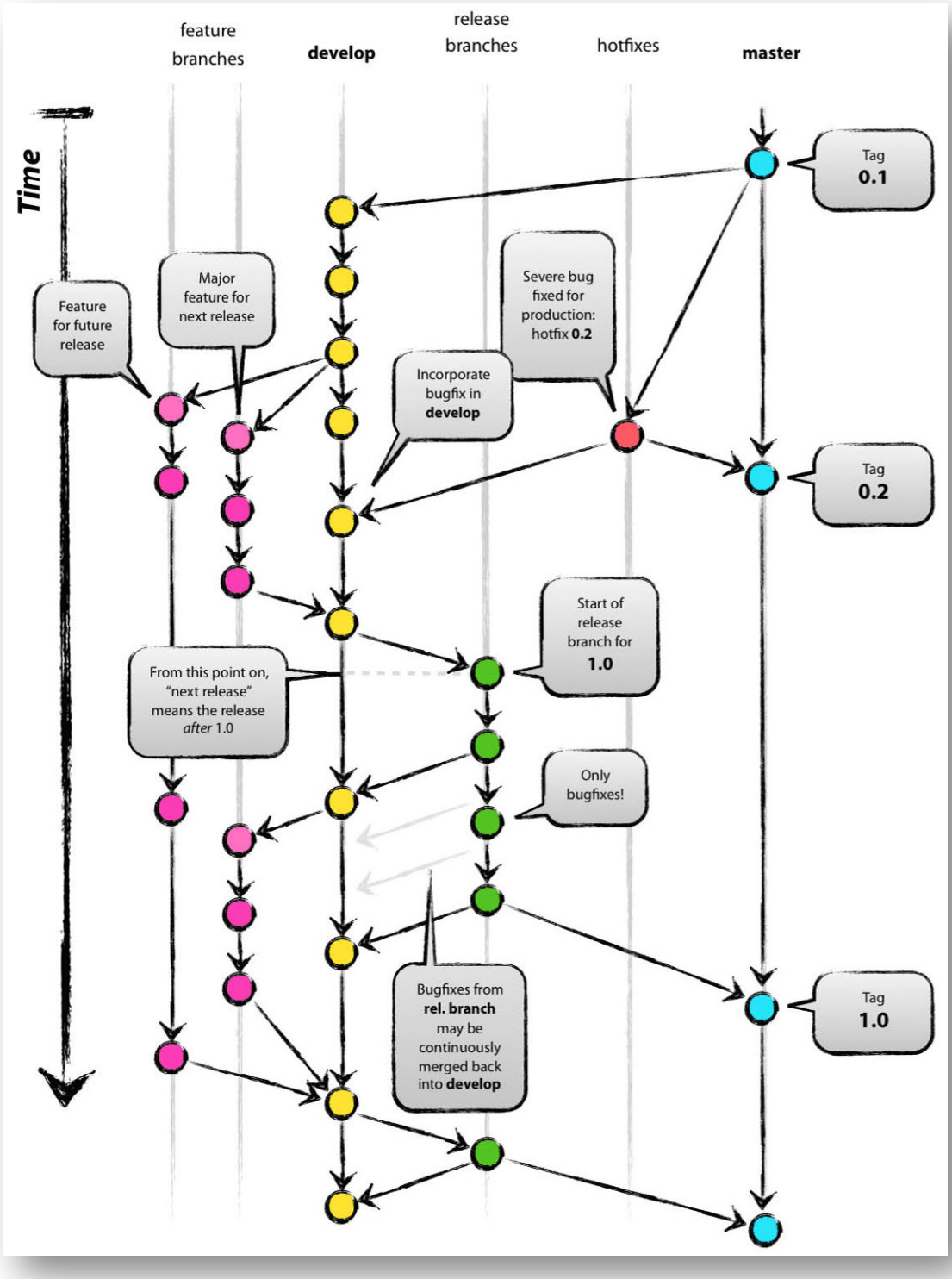


Figura 5: Flujo de trabajo de Git-Flow.

2.6 Django

Django es un framework de desarrollo de aplicaciones web escrito en Python. Es abierto y gratuito, y mantenido por la Django Software Corporation.

Django sigue el patrón modelo-vista-controlador para desarrollar aplicaciones web y se ha usado para desarrollar la API y el cliente web.

La versión de Django usada ha sido la 1.7.1, la más reciente en el momento de comenzar el proyecto. Actualmente, se encuentra en la versión 1.7.4.

Django proporciona un menú de administración de entidades a partir de la base de datos sincronizada con el sistema, lo cual hace de Django una herramienta muy potente y fácil de usar a la hora de gestionar datos.

Otra de las grandes funcionalidades que nos proporciona Django es un sistema de gestión de usuarios preparado para crear nuevos usuarios, gestionarlos y poder autenticarlos en el sistema.

2.7 Django Rest Framework

Una API REST ("REpresentational State Transfer") es un tipo de arquitectura de desarrollo web que se apoya en el estándar HTTP. Para desarrollar la API REST se ha utilizado Django Rest Framework, una librería para Django escrita en python, ya que se encuentra basado sobre el mismo sistema de vistas de Django.

Además, este framework facilita otras poderosas opciones como es una API navegable con la que se puede probar lo que estamos desarrollando, sistemas de autenticación de diferentes tipos y una gran comunidad y soporte.

Un concepto importante en REST es la existencia de recursos (elementos de información) a los que podemos acceder utilizando un identificador global.

El cliente, como un navegador por ejemplo, no necesita conocer nada de la estructura de la API más que el identificador de un recurso, es el servidor el que tiene que proveer al cliente de cualquier información que quiera conocer.

En resumen, una API REST es una librería de funciones a la que se accede por el protocolo HTTP mediante URLs en las que enviamos los datos de nuestra consulta, obteniendo como respuesta datos en formatos como XML, JSON o texto plano.

2.8 MySQL

Este sistema gestor de base de datos “open source” es uno de los más conocidos; dispone de dos versiones, “Community Edition” y “Professional Edition”, se ha utilizado la versión “community” que se distribuye bajo una licencia “GPL”, además, ofrece su utilización de forma gratuita y permite modificarlo libremente.

MySQL nos permite gestionar bases de datos relacionales, lo que nos admite crear interconexiones entre las diferentes entidades del modelo de datos.

Para poder interactuar de forma más cómoda con MySQL se ha utilizado una interfaz gráfica llamada “MySQL Workbench” que nos permite gestionar todas nuestras bases de datos mysql, realizar el diseño de ellas, ejecutar consultas, etc...

2.9 Nginx

Servidor web/proxy completamente inverso, que tiene como principal característica ser sumamente ligero y alcanzar una velocidad que nos permite servir aplicaciones web con una rapidez muy superior a la de sus competidores más directos. Es software libre, de código abierto (con una licencia FreeBSD) y multiplataforma. Es, en definitiva, un servidor web de alto rendimiento.

Capítulo 3

3 Desarrollo del proyecto

3.1 Estructura

Antes de comenzar con las fases del desarrollo del proyecto, se va a plantear una serie de conceptos necesarios para comprender la estructura del mismo.

En el proyecto se menciona el concepto de "evento" utilizado para referirse a la entidad que se encargará de almacenar los datos sobre una decisión a tomar. Éste agrupará todos los datos de título, fecha de finalización, la cual nunca puede ser inferior a 1 hora después del momento exacto de su creación, el número de votos permitido, si es un evento abierto y si las votaciones son públicas.

El "número de votos" es el número máximo de opciones que puede marcar cada participante.

El concepto de "evento abierto" hace referencia a si los participantes en un evento tienen permiso para añadir nuevas opciones de votación al mismo. Por defecto, los eventos creados en "DealtDay" no son abiertos pero el creador puede modificar esta característica si así lo desea.

El concepto "votaciones públicas" hace referencia a si se mostrará la información de quien ha votado cada opción en un evento, o si únicamemente se mostrará el número de votos totales de cada opción. Por defecto, las votaciones de los usuarios de esta aplicación no son públicas pero el creador lo puede modificar.

Dentro de la entidad "eventos" encontramos las respuestas, las cuales clasificamos en dos tipos, respuestas de tipo texto y respuestas de tipo fecha.

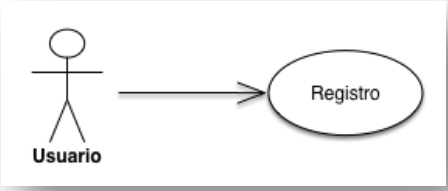
Las respuestas de tipo texto son cualquier cadena de texto que queramos almacenar; las respuestas de tipo fechas son un objeto de tipo fecha, las cuales deben de enviarse con el siguiente formato "AAAA-MM-DDThh:mm:ss-sTZD", un ejemplo sería "2014-11-11T21:02:54+01:00".

3.2 Análisis

En este apartado se van a describir las diferentes funcionalidades de la aplicación. Primero se definirá una tabla con las diferentes funcionalidades y, a continuación, se plantearán varios casos de uso que debe cumplir la aplicación.

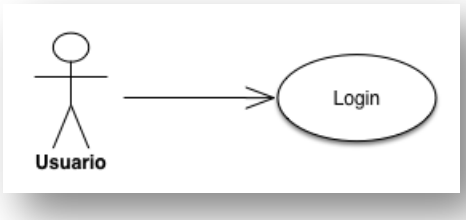
Crear un evento y poder añadir respuestas a el mismo.	Se debe poder realizar la creación de un evento por parte de un usuario a través de la aplicación web y poder añadir respuestas.
Compartir un evento con amigos.	Una vez que un usuario ha creado un evento, éste debe tener la posibilidad de compartirlo con los amigos que tenga en el sistema.
Poder añadir respuestas a un evento existente.	Cuando un evento ya ha sido creado, el usuario debe ser capaz de poder añadir nuevas respuestas a este evento.
Modo de evento abierto.	El usuario creador puede definir si en un evento cualquier persona puede añadir nuevas respuestas o, por el contrario, sólo él puede.
Modo de votaciones públicas en evento.	Un usuario debe poder definir si los votos de los diferentes usuarios que participan en el evento son anónimos o se mostrarán a los demás participantes.
Poder añadir amigos.	Un usuario debe ser capaz de enviarle una petición de amistad a un amigo.
Configurar perfil de usuario.	El sistema debe ofrecer un panel de configuración en el que el usuario puede cambiar su "nick" y su contraseña.
Crear un cuenta en el sistema.	Un usuario debe ser capaz de crear una cuenta en el sistema para poder utilizar el mismo.
Realizar una votación	Un usuario debe ser capaz de poder votar las opciones de un evento al que ha sido invitado.

3.2.1 Registro de usuario



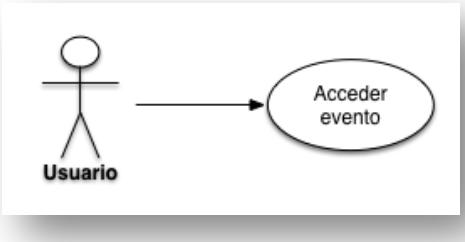
Actores	Usuario
Tipo	Primario
Propósito	Dar de alta un nuevo usuario en el sistema.
Resumen	El usuario solicita registrarse en el sistema a través de un formulario.
Precondiciones	Ninguna
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario accede a la aplicación. 2. Selecciona la opción de registrarse. 3. Rellena un formulario con el "email" y dos veces la contraseña. 7. El usuario accede a la URL indicada en el correo recibido para activar la cuenta.	4. Se comprueban que los datos enviados son válidos. 5. Se crea un usuario inactivo con los mismos datos. 6. Se envía un correo de activación al "email" facilitado por el usuario. 8. Se activa la cuenta del usuario
Curso alterno de eventos	
Paso 4	El "email" introducido es incorrecto, se devuelve un mensaje de error al usuario y se vuelve a rellenar el formulario.

3.2.2 Login de usuario



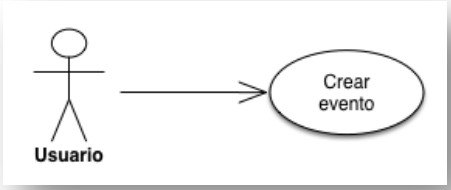
Actores	Usuario
Tipo	Primario
Propósito	Dar acceso al sistema a un usuario.
Resumen	El usuario quiere acceder a la aplicación y se le da acceso mediante un "login" con contraseña.
Precondiciones	El usuario debe estar registrado y tener su cuenta activada.
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario accede a la aplicación. 2. Rellena el formulario de "login" con su "email" y su contraseña. 3. Pulsa el botón "login".	4. Se comprueba que el "email" y la contraseña son correctos. 5. Se da acceso al usuario en el sistema y se le redirige dentro de la aplicación.
Curso alterno de eventos	
Paso 4	Si el usuario no existe o la contraseña no es válida, el usuario debe volver a rellenar el formulario.

3.2.3 Acceso a evento



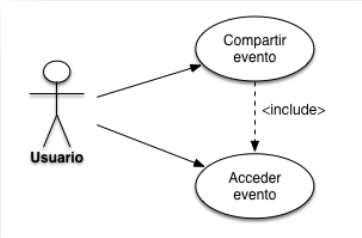
Actores	Usuario
Tipo	Primario
Propósito	Poder acceder a un evento.
Resumen	El usuario tiene que poder acceder a ver los detalles de un evento.
Precondiciones	El usuario debe estar registrado, tener su cuenta activada y participar en el evento.
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario abre a la aplicación. 2. El usuario accede a la zona “eventos”. 3. El usuario pulsa sobre el evento al que quiere acceder.	4. El sistema comprueba que el usuario participa en ese evento. 5. Se recupera la información del evento. 6. Se actualiza la información mostrada al usuario.
Curso alterno de eventos	
Paso 4	El usuario no participa en el evento seleccionado, se muestra un error al usuario.

3.2.4 Crear un evento



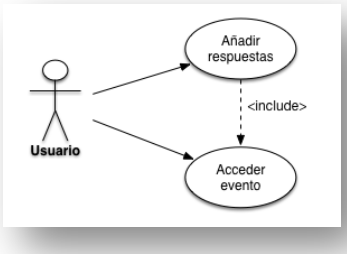
Actores	Usuario
Tipo	Primario
Propósito	Añadir un evento al sistema.
Resumen	El usuario tiene que poder configurar un evento y añadir las respuestas.
Precondiciones	El usuario debe estar registrado y tener su cuenta activada.
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario accede a la aplicación. 2. El usuario pulsa el botón para añadir un evento. 3. El usuario rellena el primer formulario que aparece, donde debe introducir el título del evento, fecha de finalización, número de votos permitidos, si el evento es abierto y si las votaciones son públicas. 4. Pulsa el botón “siguiente”. 5. Rellena el segundo formulario que aparece, donde deberá añadir el número de respuestas que desee seleccionando su tipo (texto/fecha). 6. El usuario pulsa el botón de “crear”.	7. Se comprueba que todo el formulario está rellenado correctamente. 8. Se guardan los datos en el sistema. 9. Se actualiza la información del usuario con su nuevo evento.
Curso alterno de eventos	
Paso 5	Una vez rellenado el primer formulario y pulsado el botón siguiente, se puede pulsar el botón volver para reescribir el primer formulario.

3.2.5 Compartir un evento



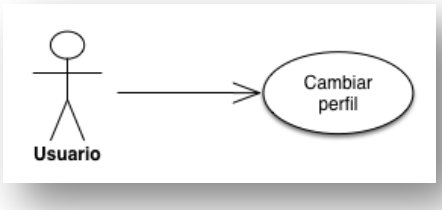
Actores	Usuario
Tipo	Primario
Propósito	Añadir un participante al evento.
Resumen	El usuario tiene que poder añadir amigos a un evento.
Precondiciones	El usuario debe estar registrado, tener su cuenta activada y pertenecer a un evento con permiso para invitar.
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario accede a la aplicación. 2. El usuario accede a los detalles de un evento. 3. Accede a la zona de "participantes". 4. Pulsa el botón "añadir". 5. Introduce el "nick" de un amigo y lo selecciona. 6. El usuario pulsa el botón de "invitar".	7. Se comprueba que el usuario seleccionado es amigo. 8. Se agrega ese amigo al evento. 9. Se actualiza la información mostrada al usuario.
Curso alterno de eventos	
Paso 7	El usuario invitado no es amigo, se muestra un mensaje de error al usuario y se vuelve a introducir otro "nick".

3.2.6 Añadir respuestas a un evento existente



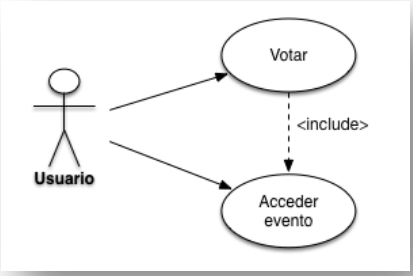
Actores	Usuario
Tipo	Primario
Propósito	Añadir respuestas a un evento.
Resumen	El usuario tiene que poder añadir respuestas a un evento ya existente.
Precondiciones	El usuario debe estar registrado, tener su cuenta activada y pertenecer a un evento con permisos para añadir respuestas.
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario abre a la aplicación. 2. El usuario accede a los detalles de un evento. 3. Accede a la zona de “respuestas”. 4. Pulsa el botón “añadir”. 5. Introduce el texto de la respuesta y su tipo (texto/fecha). 6. El usuario pulsa el botón de “añadir”.	7. Se comprueba que la respuesta es correcta y el usuario tiene permiso para realizar la acción. 8. Se agrega la respuesta al evento. 9. Se actualiza la información mostrada al usuario.
Curso alterno de eventos	
Paso 7	El usuario invitado no tiene permisos para añadir respuestas, se muestra un error.

3.2.7 Cambiar la configuración



Actores	Usuario
Tipo	Primario
Propósito	Cambiar configuración de un usuario.
Resumen	Un usuario tiene que poder cambiar su "nick" y su contraseña actual.
Precondiciones	El usuario debe estar registrado, tener su cuenta activada.
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario abre a la aplicación. 2. Accede a la zona de "perfil". 3. Rellena el formulario con los datos que desee modificar. 4. Pulsa el botón "actualizar". 5. Introduce el "nick" de un amigo y lo selecciona. 6. El usuario pulsa el botón "invitar".	7. Se comprueba que los nuevos datos son válidos. 8. Se actualiza la información en el sistema. 9. Se actualiza la información mostrada al usuario.
Curso alterno de eventos	
Paso 7	Las contraseñas introducidas son erróneas, se muestra un error y se vuelve a rellenar el formulario.

3.2.8 Realizar una votación



Actores	Usuario
Tipo	Primario
Propósito	Guardar un voto en un evento.
Resumen	Un usuario tiene que poder votar en un evento.
Precondiciones	El usuario debe estar registrado, tener su cuenta activada y participar en el evento que desea votar.
Curso típico de eventos	
Acciones de los actores	Respuestas del sistema
1. El usuario abre a la aplicación. 2. Accede a los eventos en los que participa. 3. Selecciona el evento en el que quiere votar. 4. Selecciona las respuestas que desea votar. 5. Pulsa el botón “votar”	6. Se comprueba que los votos son válidos. 7. Se guarda la información en el sistema. 8. Se actualiza la información mostrada al usuario.
Curso alterno de eventos	
Paso 6	Los votos introducidos son erróneos, se muestra un mensaje de error al usuario y se procede a volver a votar.

3.3 Diseño

En esta sección se va a definir el modelo de datos creado para este proyecto y se mostrarán diferentes "wireframes" sobre los que se ha creado la aplicación mediante la información recogida en la fase de análisis.

3.3.1 Modelo de datos

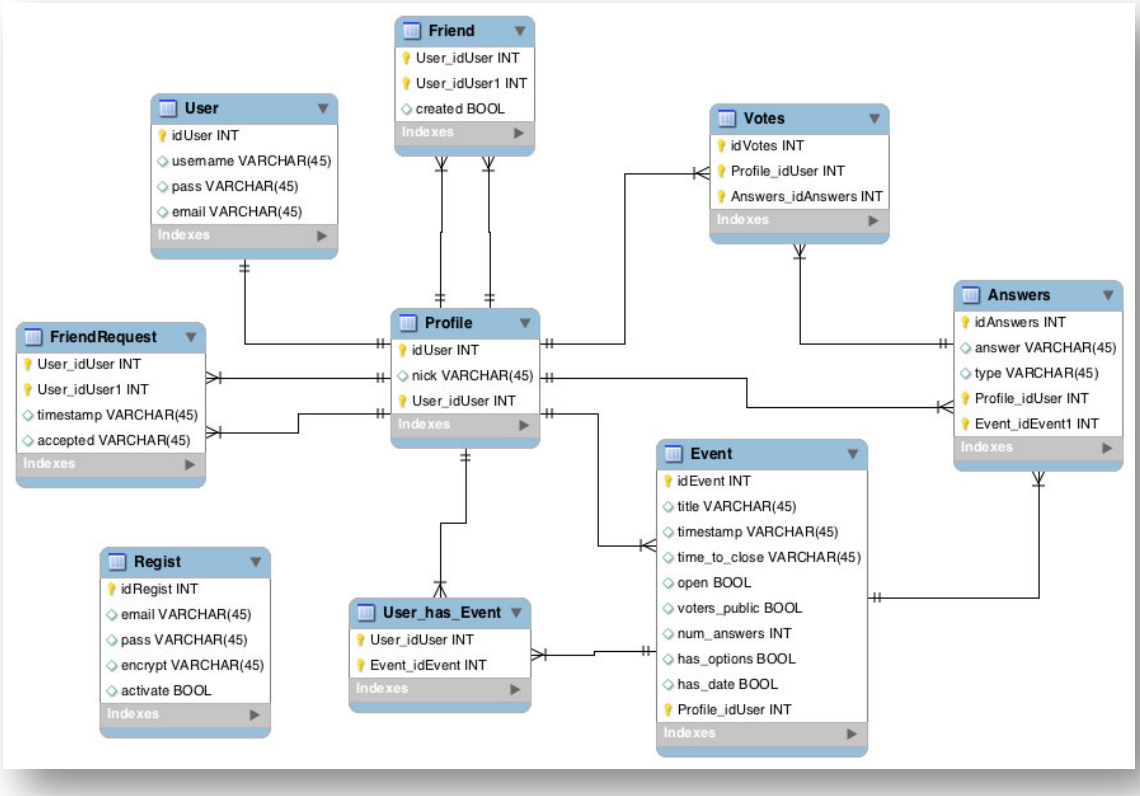


Figura 6: Modelo de datos de la aplicación.

3.3.2 Tablas

A continuación se definirán las tablas que conforman la base de datos del sistema. Se han obviado las columnas 'id', de clave única, de cada una de las tablas.

Tabla	User
Columnas	Username, password, email
Claves foráneas	No tiene
Esta tabla la proporciona el sistema de Django (framework utilizado para el desarrollo del proyecto) para facilitar la gestión de los usuarios. Aunque está incompleta, su uso es necesario y la finalidad que tiene que cumplir no necesita de más campos. Para completar esta tabla, se ha creado una tabla Profile a modo de extensión.	

Tabla	Profile
Columnas	Nick
Claves foráneas	User_id
La tabla Profile está pensada como una extensión de la tabla User para almacenar datos propios de la aplicación, como es el nick del usuario, y que nos servirá para poder almacenar más datos en el futuro. Estos datos van en una tabla aparte para no romper el esquema de una tabla de usuario básica y con pocos datos. Además, evita tener que reescribir el modelo de usuario provisto por el entorno Django, que podría dar lugar a numerosos errores.	

Tabla	Regist
Columnas	Email, password, encrypt, activate
Claves foráneas	No tiene
Mediante la tabla Regist se gestiona el registro de usuarios desde el momento en que se registran hasta que su cuenta se activa. La tabla es igual que User, pero con las columnas encrypt y activate añadidas. La idea es que, tras registrarse un usuario, se le envíe un correo electrónico con una URL de activación. Hasta que no se haya activado la cuenta, no se activa el registro creado en User. "Encrypt" sirve para controlar que no haya activaciones falsas. Se trata de un token asignado a un usuario, una clave hash que nos permitirá autenticar la URL de activación (junto con el correo del usuario encriptado). "Activate" sirve para controlar que el usuario ya tiene un User activo.	

Tabla	Event
Columnas	Title, timestamp, time_to_close, open, voters_public, num_answers, has_options, has_date
Claves foráneas	Profile_id

Esta tabla es en gran parte el centro de la base de datos. Es la utilizada para guardar toda la información de un evento.

Está compuesta por el campo "title" donde se almacena el título del evento, el campo "timestamp" que se genera automáticamente cuando se crea un evento y almacena la fecha de su creación, el campo "open" que especifica si un evento es abierto, el campo "voters_public" que almacena si las votaciones de un evento son públicas y el campo "num_answers" que guarda el número de respuestas que un participante puede votar.

También se encuentran dos campos especiales que son "has_options" que guardan un valor booleano especificando si un evento tiene respuestas de tipo "texto", y el campo "has_date" que guardan un valor booleano especificando si un evento tiene respuestas de tipo "fecha".

Por último, se encuentra una clave foránea que apunta a un Profile_id donde se almacena el Profile del creador del evento.

Tabla	Answers
Columnas	Answer, type
Claves foráneas	Profile_id, Event_id

Esta tabla almacena la información de todas las respuestas asociadas a un evento.

En ella se encuentra el campo "answer", que almacena el valor de la respuesta, y el campo "type" donde se almacena el tipo de la respuesta. Este último campo sólo puede tener dos valores que son "TX" para respuestas de tipo texto y "DT" para respuestas de tipo fecha.

También se encuentra una clave foránea a Profile_id donde se almacena el creador de una respuesta, y la clave foránea Event_id que almacena a qué evento pertenece una respuesta.

Tabla	Votes
Columnas	No tiene
Claves foráneas	Profile_id, Answer_id

Esta tabla almacena la información de todos los votos que hay en el sistema.

En ella se encuentra una clave foránea a Profile_id, donde se almacena el dueño de un voto, y la clave foránea Answer_id que almacena la respuesta a la que pertenece el voto.

Tabla	User_has_Event
Columnas	No tiene
Claves foráneas	Profile_id, Event_id

Esta tabla almacena la información sobre los usuarios que participan en los diferentes eventos.

En ella se encuentra una clave foránea a Profile_id donde se almacena el usuario que pertenece al evento y la clave foránea Event_id que almacena el evento en cuestión.

Tabla	Friend
Columnas	Created
Claves foráneas	Profile_id, Profile_id_1

Esta tabla almacena la información de los amigos de un usuario.

Encontramos el campo "created" que se autogenera cuando se crea un amigo nuevo y almacena la fecha y hora de la creación.

En ella se encuentran las claves foráneas "Profile_id" y "Profile_id_1" que almacenan el Profile de los dos usuarios que son amigos.

Tabla	FriendRequest
Columnas	Timestamp, accepted
Claves foráneas	Profile_id, Profile_id_1

Esta tabla almacena la información de las peticiones de amistad que envía un usuario.

Encontramos el campo "timestamp", que se autogenera cuando se crea una petición de amistad nueva y almacena la fecha y hora de la creación, y el campo "accepted", que almacena un valor "booleano" que refleja si la solicitud de amistad ha sido aceptada o no.

En esta tabla se encuentran las claves foráneas "Profile_id" que almacenan la información del Profile del usuario que envía la petición de amistad, y "Profile_id_1" que almacenan la información del Profile del usuario que recibe la petición de amistad.

3.3.3 Wireframes

Antes de comenzar con el desarrollo, se crearon algunos wireframes para tener una idea más clara de cómo iba a ser la herramienta y poder tener una mejor visión de las funcionalidades que se debían implementar.

Un wireframe de un sitio web es un esquema, una guía visual que representa la estructura del sitio.

Los wireframes mostrados a continuación se han diseñado mediante el software Sketch 3.

3.3.3.1 Pantalla de acceso

Figura 7: Wireframe pantalla de acceso.

En esta imagen se muestra el formulario básico de acceso con los campos de email y password y el botón de login.

Las pantallas de registro y recuperar contraseña serían igual solo que con los campos de formulario necesarios en cada caso.

3.3.3.2 Estructura y menú de la aplicación

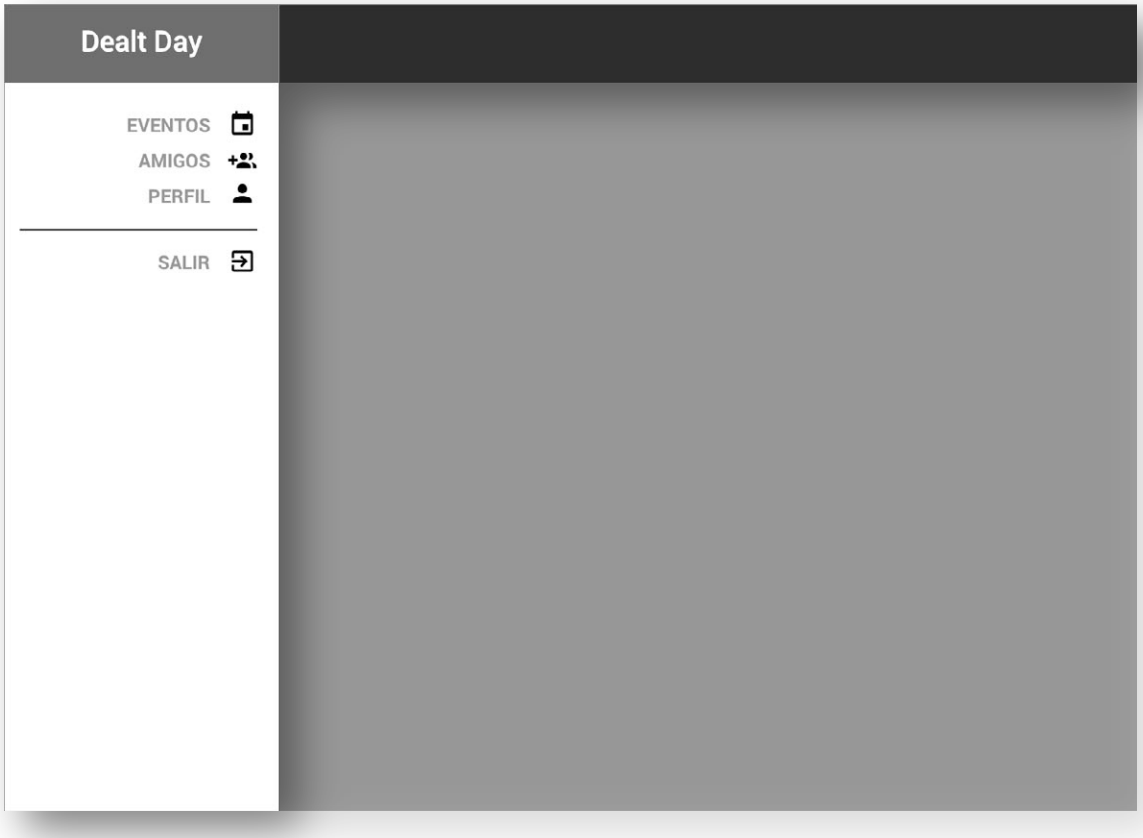


Figura 8: Wireframe estructura y menú aplicación.

En esta imagen observamos cómo se muestra la estructura y el menú de la aplicación para una pantalla mayor a 960 píxeles; si es menor, la estructura varía y el menú se oculta pudiendo desplegarse a través de un botón.

En el resto de la pantalla, que aparece en gris en la imagen, se mostrará el contenido de cada sección que aparece reflejada en el menú, es decir, “eventos”, “amigos” y “perfil”.

3.3.3.3 Estructura amigos

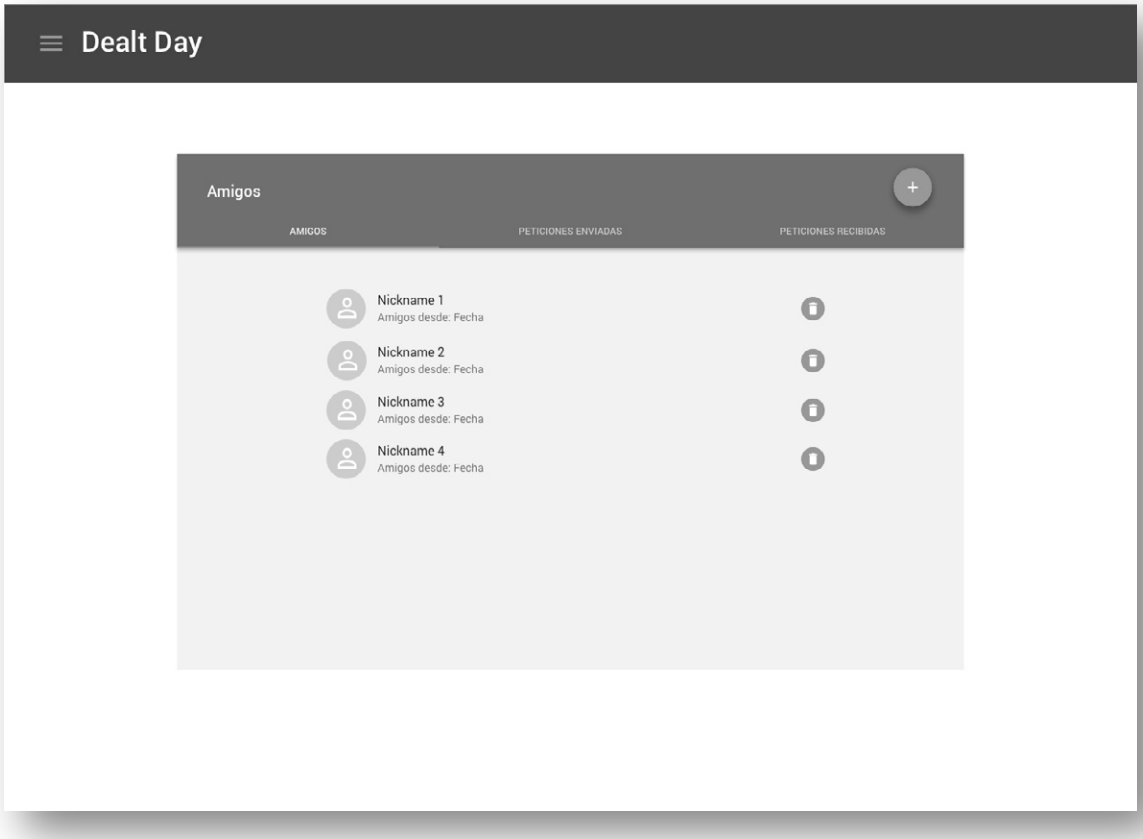


Figura 9: Wireframe de la sección amigos.

En esta figura se observa la estructura de la sección amigos. Este elemento se encuentra organizado en tres pestañas: una para los amigos, una para las peticiones de amistad enviadas y otra para las peticiones de amistad recibidas.

También podemos observar un botón redondo en la parte superior que nos servirá para desplegar la opción de enviar las peticiones de amistad.

3.4 Implementación

Con los datos obtenidos en los apartados anteriores se ha procedido a realizar la implementación del proyecto. Se ha dividido en dos partes claramente diferenciadas; por un lado, el desarrollo de la parte del servidor o "backend" y, por otro lado, la parte de desarrollo del cliente web o "frontend".

3.4.1 Desarrollo API REST

3.4.1.1 Herramientas utilizadas

Como entorno de desarrollo (IDE) se ha utilizado PyCharm, un entorno que proporciona análisis de código, depurador gráfico, realización de tests unitarios y soporta el desarrollo web con Django. Está desarrollado por JetBrains y se usa para la programación en Python. Para este proyecto se ha usado la versión 4.0.

La depuración del código y las pruebas se han llevado a cabo mediante la herramienta Postman, un cliente REST que funciona como extensión del navegador Google Chrome.

Postman no permite realizar peticiones HTTP totalmente configurables añadiendo distintos tipos de autenticación y cabeceras a cada petición.

Para el mantenimiento y creación de la base de datos se ha usado MySQL Workbench, una herramienta visual de diseño de bases de datos orientada al uso de bases de datos de tipo MySQL.

3.4.1.2 Recursos

A partir de las funcionalidades del sistema ya definidas y del modelo de datos, se van a explicar los diferentes recursos creados para proveer de funcionalidad a nuestra API. En el apéndice de este documento encontraremos una clasificación mucho más detallada de cada recurso y sus distintas funcionalidades.

Regist	Este recurso está asociado a la tabla Regist que se encarga de almacenar los datos de usuario después de que éste haya hecho la petición de registrarse en el sistema. Este recurso sólo requiere de la operación de creación (POST) ya que únicamente se usa para añadir filas a la tabla Regist. Una vez hecha la petición, si es correcta, el sistema se encarga de enviar un correo electrónico al futuro usuario para llevar a cabo la activación de la cuenta.
Login/Logout	El recurso Login se asocia con la tabla User, aunque no llega a modificarla. Sirve para iniciar sesión mediante una llamada POST y para finalizarla mediante una llamada GET. La petición para hacer "login" también sirve para proporcionar un "Token" o clave de acceso a dispositivos móviles que vayan a hacer uso de la API.
Forgot-password	Este recurso nos permite, mediante una petición POST, enviar un correo para recuperar nuestra contraseña.
Change-password	Este recurso nos permite, mediante una petición POST, cambiar nuestra contraseña de acceso al sistema.
Me	Este recurso maneja la operación GET para poder recuperar toda la información relacionada con un usuario.
Event	El recurso Event nos va a permitir realizar todas las tareas correspondientes a la gestión de un evento, como son sus operaciones de creación, listado, eliminación, actualización y ver el detalle de un evento.
Answer	Al igual que el recurso Event, este recurso nos permite realizar la misma gestión pero sobre la entidad Answer.
Friends	Este recurso está orientado a que, en una sola petición, se puedan obtener todos los amigos que tiene un usuario y también permite la eliminación de un amigo.
Share	El recurso Share nos permite, mediante una sola petición, añadir a un amigo a que participe en un evento. También nos permite ver todos los eventos en los que estamos participando.
Friend_request	Este recurso nos permite crear solicitudes de amistad mediante una petición POST, ver todas las peticiones de amistad gestionadas por un usuario (tanto recibidas como enviadas) mediante una petición GET, aceptar una petición de amistad mediante una petición PATCH o eliminarla con un DELETE.
Vote	Este recurso nos permite realizar votaciones sobre un evento, eliminar un voto y ver todos los votos que hemos emitido sobre un determinado evento.

3.4.1.3 La API

A partir de los recursos de API REST construidos y usados en la aplicación, se ha documentado la API de forma que sea entendible y fácil de manejar para cualquier persona que quiera colaborar en el desarrollo del proyecto.

Cada método de la API consta de una pequeña definición sobre cuál es su funcionalidad y una serie de tablas explicativas que se detallan a continuación.

La primera tabla nos muestra información sobre cómo realizar la petición al servidor.

Request URL	http://127.0.0.1:8000/api/login/
Request Type	POST
Request Format	JSON
Request Authetication	None

- **REQUEST URL:** dirección a la que se realiza la petición.
- **REQUEST TYPE:** tipo de petición, ya sea GET, POST, PUT, PATCH o DELETE.
- **REQUEST FORMAT:** formato en el que se debe enviar la petición, sólo en el caso de que sea una petición en la que se envían datos. Se ha usado JSON como formato estándar.
- **REQUEST AUTHENTICATION:** tipo de autenticación necesaria para realizar la petición. En el caso de esta API, se ha usado la autenticación de Sesión para los clientes web y un Token, que es una clave única para cada usuario que lo permite autenticarse para el uso de la API en dispositivos móviles.

La segunda tabla nos muestra cómo debemos enviar los datos al servidor en caso de que éstos sean necesarios (POST, PUT y PATCH).

Example Request

{
“email”: EMAIL,
“password”: PASSWORD
}

La tercera tabla es el tipo de respuesta esperado que incluye el código de estado que debería devolver la petición en caso de que todo haya ido correctamente y un ejemplo de cómo se mostrarán los datos de la respuesta en formato JSON

HTTP 200 OK

{
“email”: EMAIL,
“token”: TOKEN
}

Por último, la cuarta tabla muestra los errores que puede devolvernos el sistema al realizar la petición, junto con su detalle.

Nombre	Detalle
“ErrorLogin”	"Password o usuario incorrectos."
“ErrorLogin”	"El usuario aún no ha sido activado".

3.4.1.4 Seguridad

En la actualidad, la seguridad es un tema primordial a la hora de desarrollar aplicaciones. Garantizar que los datos de un usuario estén bien protegidos es primordial, al igual que proteger la API REST de posibles ataques informáticos. Gracias a la combinación de Django y Django Rest Framework podemos aplicar las medidas de seguridad necesarias para garantizar un alto nivel de seguridad.

Django nos permite cifrar todas las contraseñas de los usuarios. Por defecto, este “framework” usa el sistema “PBKDF2” (Password-Based Key Derivation Function 2) que es el sistema recomendado por el “NITS” (Instituto Nacional de Estándares y Tecnología), aunque Django también nos permite cambiar este sistema por otros que él mismo nos ofrece.

También nos ofrece varios métodos, como son las variables de sesión, para autenticar a los usuarios en el sistema. Django realiza esta labor a través de una “cookie”, llamada “sessionid”, que se almacena en el navegador y es protegida a través de “HTTPOnly” lo que la preserva de ser accedida desde el lado del cliente. Esta protección también se puede desactivar en Django aunque no es recomendable.

Gracias a Django Rest Framework podemos incluir otros tipos de autenticación como el sistema basado en Tokens. Este sistema es una cadena de caracteres que identifica de forma única a un usuario en el sistema. Para poder aplicar este tipo de autenticación debemos incluir el Token en la cabecera de la petición http añadiendo lo siguiente:

```
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Cada recurso de nuestra API lo podemos proteger de manera diferente y totalmente configurable gracias a la definición de los permisos necesarios para poder hacer uso del recurso que deseemos usar. Un ejemplo de aplicación seria el siguiente:

```
class Event(ModelViewSet):

        permission_classes = (IsAuthenticated,)
```

En “permission_classes” definiríamos los diferentes permisos necesarios para usar cada recurso. “IsAuthenticated” es el método por defecto de Django Rest Framework que se ha configurado para que use autenticación basada en sesión o basada en Token.

Django Rest Framework también nos permite especificar qué tipo de peticiones HTTP están disponibles en cada recurso. En el ejemplo anterior, permitimos todas las peticiones sobre el recurso; sin embargo, en el siguiente ejemplo únicamente vamos a permitir la operación POST sobre el recurso:

```
class Event(GenericViewSet, CreateModelMixin):

        permission_classes = (IsAuthenticated,)
```

3.4.2 Desarrollo cliente web

3.4.2.1 Herramientas utilizadas

Como entorno de desarrollo (IDE) también se ha utilizado PyCharm ya que también nos da soporte para desarrollar aplicaciones web, y también añade soporte para utilizar Grunt.

Para probar y visualizar la aplicación web se han usado los navegadores Google Chrome, Mozilla Firefox y Safari.

Como complemento al desarrollo del cliente web se ha usado una extensión de "Google Chrome" llamada "Batarang" que nos permite depurar distintas funcionalidades de AngularJS.

3.4.2.2 MVC

El modelo-vista-controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello se construyen tres componentes distintos: el modelo, la vista y el controlador.

AngularJS aplica este tipo de patrón pero con alguna peculiaridad. En primer lugar, surge un nuevo actor hacer llamado "Scope" que permite la comunicación entre la vista y el controlador.

Otra peculiaridad es que el modelo vive dentro del controlador.

A continuación vamos a ver una imagen del patrón MVC clásico y otra del patrón MVC que se aplica en AngularJS.

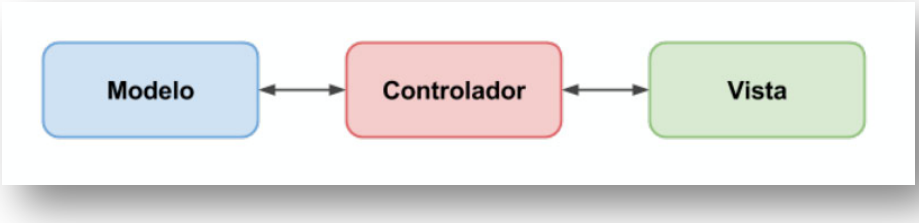


Figura 10: Patrón MVC clásico.

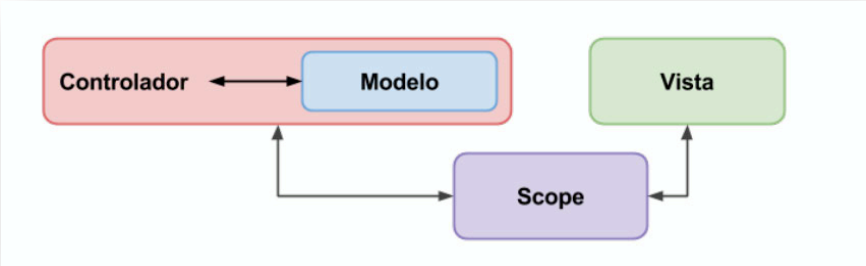


Figura 11: Patrón MVC AngularJS.

El patrón funciona de la siguiente manera:

- **Modelo:** son los datos que, en este caso, recibimos de la API REST en formato "JSON".
- **Vista:** son los templares HTML y las directivas de AngularJS.
- **Controlador:** son los controladores, servicios, etc., que permiten interactuar con la aplicación web.

3.4.2.3 Comunicación cliente-servidor

Para poder comunicar a un cliente web con el servidor se ha utilizado una tecnología "AJAX".

AJAX (Asynchronous Javascript And XML, Javascript asíncrono y XML) es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente (el navegador web) mientras se mantiene la comunicación asíncrona en segundo plano con el servidor. Así, se puede actualizar y realizar cambios sobre páginas sin necesidad de recargarlas. Esto mejora la velocidad, navegación y usabilidad de la aplicación web.

Las llamadas de AJAX se efectúan en Javascript, XMLHttpRequest es el objeto que se utiliza para intercambiar datos con el servidor y XML es el formato que se usa generalmente para la transferencia de datos aunque se pueden usar otros como JSON, que ha sido la elección de este trabajo.

En el proyecto se ha utilizado este tipo de peticiones a través de dos servicios que nos ofrece AngularJS llamados "$http" y "$resource".

"$http" es un servicio del núcleo de AngularJS que nos facilita la comunicación con un servicio remoto y, por su parte, "$resource" nos permite lograr lo mismo que a través de "$http" pero facilitando un nivel de abstracción mayor.

Un ejemplo de cómo realizaríamos una petición GET con el servicio "$http" sería el siguiente.

```
// El campo "URL" es a donde se realiza la petición

$http.get(URL).success(function(data){

// Aquí recibimos los datos de respuesta

}).

error(function (error){

// Aquí podemos realizar el manejo de errores

});
```

3.4.2.4 Diseño web

El diseño web es la planificación, diseño e implementación de un sitio web. Requiere tener en cuenta la navegabilidad, interactividad, usabilidad, arquitectura de la información y la interacción de los medios.

A partir de los "wireframes" realizados en la fase de diseño, se ha implementado la aplicación web tratando de seguir un diseño simple y navegable.

El menú que encontramos en la parte izquierda de la pantalla nos permitirá visualizar los diferentes contenidos de la web que se detalla a continuación:

- **Inicio:** donde encontraremos unas simples instrucciones de cómo poder hacer uso de la aplicación.
- **Eventos:** núcleo de la aplicación y espacio donde podremos ver el listado de todos los eventos, acceder a los detalles de uno y crear eventos nuevos.
- **Amigos:** esta sección incluye la gestión de amigos y de las solicitudes de amistad.
- **Perfil:** desde donde podemos cambiar los ajustes de la cuenta de un usuario registrado.

3.4.2.5 Seguridad

Al igual que en la parte del servidor, la parte del cliente también debe implementar diferentes medidas de seguridad para facilitar el trabajo en el servidor ahorrando peticiones innecesarias o accesos no deseados.

Una de las medidas de seguridad adoptadas es que todos los formularios de la aplicación se encuentran protegidos y no permite el envío de datos al servidor hasta que los datos introducidos son correctos.

Otra medida implementada es que al solicitar los plantillas al servidor, Django comprueba si el usuario está autenticado con un método como el siguiente.

```
def index(request):
        if request.user.is_authenticated()
                return render(request, 'index.html')
        else:
                return redirect('/login')
```

3.5 Pruebas

A continuación se van a describir algunas de las pruebas que se han realizado sobre la aplicación web ya desplegada y corriendo en el servidor de Amazon.

3.5.1 API

Las pruebas de la API se han hecho con ayuda de la extensión del navegador Google Chrome Postman, un cliente de REST que facilita probar diferentes recursos mediante peticiones HTTP.

Las pruebas han consistido en realizar peticiones a todos los recursos desarrollados para comprobar su correcto funcionamiento.

Se crearon varios usuarios con "Token" asignado y se han hecho pruebas con diferentes métodos de autenticación como "Token", autenticación básica y sin autenticación.

Cada operación CRUD (Create, Read,Update, Delete) dentro de un recurso, como se puede comprobar en la documentación de la API, tiene una respuesta esperada y una tabla de errores posibles. Las pruebas realizadas sobre cada recurso han consistido en el envío de datos erróneos esperando recibir un error concreto y el envío de datos correctos para obtener una respuesta positiva.

Todas las pruebas realizadas se han desarrollado de manera incremental; es decir, cada vez que se desarrollaba una funcionalidad, se probaban todas las funcionalidades relacionadas con ella para comprobar que todo seguía funcionando de manera correcta.

Tener una separación clara de los recursos y los "endpoints" que se tienen que probar ofrece la facilidad de realizar pruebas de caja negra, donde sólo hay que preocuparse por las entradas y salidas que recibe, lo que da robustez y mantenibilidad al sistema.

3.5.2 Registro

Probamos el registro de usuario y el login de la aplicación. En el registro, los validadores, tanto de JavaScript como los que validan desde el servidor, funcionan correctamente.

Una vez completado el registro con campos válidos, se debería recibir un correo de confirmación con un enlace que, al usarlo, nos redirija a la aplicación con la cuenta ya validada.

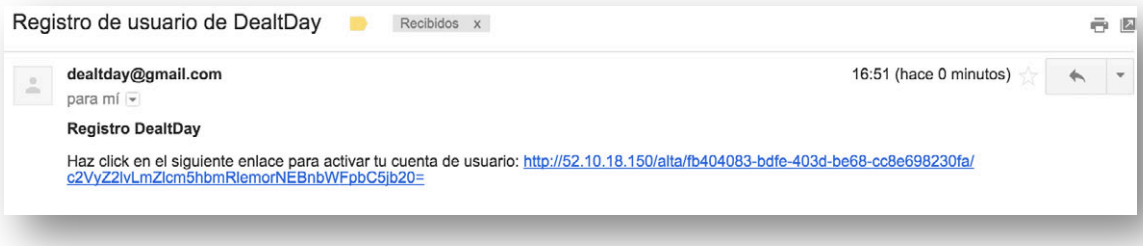


Figura 12: Email de registro recibido.

El correo de confirmación se recibe correctamente y la cuenta se activa al visitar el enlace que se adjunta. Esto se sabe porque, desde la vista de "login" y con las credenciales correctamente insertadas, al pulsar el botón "login", el usuario se redirige a la parte principal de la aplicación.

Además, como se muestra en la siguiente imagen, el servidor envía al navegador las "cookies" de sesión y un "token" llamado "csrftoken". Este "token" es una clave aleatoria que envía Django para, posteriormente, evitar ataques de tipo CSRF (Cross Site Request Forgery), en el que comandos no autorizados son transmitidos por un usuario en el que el sitio Web confía.

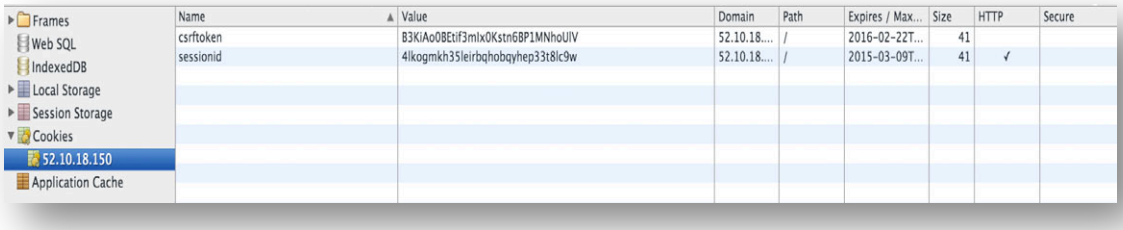


Figura 13: "Cookies" navegador.

3.5.3 Rendimiento

Una de las pruebas realizadas ha consistido en confirmar que la sección de eventos, núcleo principal de la aplicación, se carga en un tiempo considerable. Esta sección es la que más se va a visitar durante el uso de la aplicación, por lo que su funcionamiento y rapidez son esenciales.

A continuación, se muestra una tabla de pruebas de carga de la página principal en varios navegadores (en milisegundos):

Chrome	Firefox
263.900	240.000
265.350	258.000
275.060	245.000
275.540	249.000
298.560	243.000
264.500	249.000
263.700	247.000
273.910	282.000
364.740	244.000
479.100	250.000

A la vista de los resultados obtenidos, podemos concluir que la aplicación responde con la suficiente rapidez a las peticiones realizadas por los usuarios.

3.6 Despliegue

3.6.1 Creación de la instancia

Una vez finalizado el desarrollo del API y la aplicación Web, se decidió probar la aplicación fuera del entorno local del ordenador donde se había creado, por lo que se decidió probar Amazon Web Services (AWS). Amazon nos ofrece la posibilidad de obtener una instancia gratuita durante el primer año.

Lo primero que debemos hacer antes de lanzar una nueva instancia es definir los grupos de seguridad, es decir, configurar los puertos por donde podremos acceder al servidor.

Se configuraron los puertos 22, 80 y 587 del protocolo TCP/IP. El puerto 22 se abre para que podamos tener acceso a la máquina mediante el protocolo SSH. Con el puerto 80 abierto permitimos el acceso mediante HTTP, protocolo usado por los navegadores Web. Por último, el puerto 587 lo utilizamos para enviar correos mediante una cuenta de Gmail.

Después se creó una clave de acceso ("KeyPair") que nos sirvió como credencial para acceder a la máquina vía SSH. Con esto, tendremos un archivo de extensión .pem que contiene una clave privada cifrada en RSA.

Una vez completados estos pasos, se puede lanzar la instancia. En nuestro caso, la instancia es una imagen Ubuntu Server 14.04 sobre un sistema de 64 bits. Cuenta con un VCPU (virtual CPU, una CPU física asignada a una máquina virtual por el proveedor del servicio) y una memoria RAM de 1024 MB.

El último paso antes de lanzar la instancia es configurar la seguridad. Se asigna el grupo de seguridad creado con los puertos 22, 80 y 587 abiertos.

Una vez completados estos pasos, se puede lanzar la instancia.

3.6.2 Instalación de la aplicación

Estas son las tecnologías que vamos a utilizar para instalar la aplicación de Django:

- **Nginx:** servidor web/proxy completamente inverso, que tiene como principal característica ser sumamente ligero y una velocidad que nos permite servir aplicaciones web con una rapidez muy superior a la de sus competidores más directos. Es software libre, de código abierto (con una licencia FreeBSD) y multiplataforma. Es, en definitiva, un servidor web de alto rendimiento.
- **uWSGI:** interfaz encargada de comunicar Nginx con el intérprete de Python.

- **MySQL:** gestor de bases de datos que utilizaremos para crear y administrar la base de datos de la aplicación.
- **Virtualenv:** herramienta de Python usada para crear entornos aislados sin interferir con otros entornos ni con los paquetes del sistema de Python.

A continuación mostramos la configuración de Nginx:

```
server {
        listen 80;
        server_name 52.10.18.150;
        error_log /var/apps/dealday/log/nginx.error.log;
        access_log /var/apps/dealtday/log/nginx.access.log;
        index index.html;

        location / {
                expires -1;
                root /var/apps/dealtday/dealtday;
                include uwsgi_params;
                uwsgi_pass unix:///var/run/uwsgi/app/dealtday/socket;
        }

        location /static {
                autoindex on;
                alias /var/apps/dealtday/static/;
                expires -1;
        }
}
```

La directiva listen, en el bloque de servidor, le dice a Nginx el puerto TCP en el que debe escuchar las conexiones HTTP. En este caso y, normalmente, es el puerto 80.

La directiva "server_name" permite al administrador proporcionar un "hosting" basado en un nombre virtual. Esto permite que varios dominios puedan ser servidos desde una sola dirección IP. En este caso sólo queremos servir un dominio e indicamos la IP del servidor.

Las directivas "error_log" y "access_log" indican la directiva de los "logs" de acceso y errores de Nginx.

Las directivas "location" nos permiten configurar cómo Nginx responderá a las peticiones de recursos dentro del servidor, cubriendo peticiones para archivos y directorios específicos. En este proyecto, especificamos dónde se encuentra la aplicación Django y los archivos estáticos.

A continuación veremos la configuración de uwsgi:

```
[uwsgi]
home = /var/apps/dealtday
chdir = /var/apps/dealtday/dealtday
plugins = python
env = DJANGO_SETTINGS_MODULE=dealtday.settings.production
module = dealtday.wsgi:application
logto = /var/apps/dealtday/log/uwsgi.log
virtualenv = /var/apps/dealtday/env
socket = /var/run/uwsgi/app/dealtday/socket
```

Algunas de las configuraciones importantes que le estamos indicando son:

- **Chdir**: ruta donde se encuentra la aplicación Django dentro del servidor.
- **Env**: indicamos las variables de entorno. En nuestro caso la variable DJANGO_SETTINGS_MODULE, que señala qué archivo debe usar la aplicación como archivo de configuración de Django.
- **Module**: la ruta del archivo de configuración Wsgi en nuestro proyecto.
- **Virtualenv**: ruta donde se encuentra el entorno virtual de Python que queremos usar.

Con estas configuraciones creadas correctamente ya podría correr una aplicación Django en el servidor, pero aún son necesarios algunos pasos para que DealtDay lo haga correctamente.

Se ha creado un entorno virtual de Python y se han instalado en él las librerías de Python necesarias para un correcto funcionamiento. Comprobamos las librerías instaladas para saber si falta alguna con el comando:

```
pip freeze
```

```
Django==1.7.1
MySQL-python==1.2.5
argparse==1.2.1
djangorestframework==2.4.3
pytz==2014.9
wsgiref==0.1.2
```

Después, mediante MySQL, creamos una base de datos vacía pero con el mismo nombre y contraseña que hemos indicado en el archivo settings de Django. Esto es:

```
CREATE SCHEMA `dealtday DEFAULT CHARACTER SET utf8;
```

Por último, desde la carpeta que contiene el archivo manage.py de nuestra aplicación, hay que ejecutar el comando

```
python migrate –-settings=dealtday.settings.production
```

para sincronizar la base de datos con la aplicación. La ejecución del comando se encargará de crear las tablas que necesita la aplicación y nos dará la posibilidad de crear un usuario administrador del sistema.

Con las configuraciones anteriores hechas correctamente, y reiniciando los servicios de Nginx y uwsgi, la aplicación ya debe estar corriendo en el servidor

```
service nginx restart
service uwsgi restart
```

Capítulo 4

4 Conclusiones

El objetivo de este proyecto fue poder facilitar la toma de decisiones y organización a un grupo de personas.

Para lograr el objetivo se ha desarrollado un sistema que nos permite realizar dos tipos de funcionalidades, la primera es la encargada de gestionar la creación de los eventos, las votaciones de las opciones de cada evento y la invitación de otros usuarios para participar en el evento; por otro lado, encontramos la gestión de los amigos, que nos permite enviar y recibir peticiones de amistad y gestionar los amigos que actualmente tenemos en el sistema.

Para ello se implementó una API REST mediante Django y Django Rest Framework, que nos permite interaccionar con el sistema tanto desde una aplicación web de escritorio como desde un dispositivo móvil; también se desarrolló una aplicación web adaptable a distintos dispositivos y resoluciones, que permite hacer uso de la API mediante el las tecnologías HTML5, CSS3 y Javascript.

Las distintas tecnologías utilizadas para el desarrollo considero que han sido totalmente adecuadas, ya que me han permitido alcanzar todos los objetivos propuestos al comienzo del proyecto, además de haber adquirido conocimientos en distintos “frameworks” muy potentes.

A nivel personal creo que este desarrollo me ha aportado una visión más amplia del desarrollo y la gestión de un proyecto completo, aumentando mi experiencia y conocimientos en el mundo de la programación web.

Una vez finalizado el proyecto creo que tiene un gran potencial y escalabilidad, listo para ser distribuido una versión “beta” del mismo y poder ser probado en un entorno real.

Capítulo 5

5 Futuros proyectos

Para el progreso de DealtDay, se plantea la posibilidad de integrarla con las distintas redes sociales (Facebook, Twitter, Google, etc.) aprovechando el masivo uso que actualmente se hace de ellas. Este proyecto facilitaría en gran medida el uso de la aplicación, ya que se podría ahorrar el registro y login, y también obtener los amigos que usen la aplicación a través de la misma red social.

Otro punto de mejora sería poder implementar, en un segundo plano, un gestor de tareas que se encargue de realizar labores como eliminar los usuarios que se han registrado y no han activado su cuenta, poder enviar a varios amigos a la vez la petición de participar en un evento o se encargue de enviar todos los correos para confirmar la invitación.

También se propone seguir actualizando el "framework" utilizado para desarrollar el aspecto visual de la aplicación ya que, con las mejoras que se derivan de las actualizaciones que se vayan realizando, puede resultar interesante implementarlas.

Capítulo 6

6 Bibliografía

- Azaustre, Carlos. **Automatizar tareas en JavaScript con Grunt.js** [última consulta: 12/01/2015] <http://carlosazaustre.es/blog/automatizar-tareas-en-javascript-con-grunt-js/>

- Christie, Tom. **API Guide** [última consulta: 10/02/2015] <http://www.django-rest-framework.org/>

- Cloud Techie. **How to install and configure Nginx on Amazon EC2 RHEL and Ubuntu instantes.** [última consulta: 12/02/2015] <http://www.comtechies.com/2014/03/How-to-install-and-configure-nginx-on-amazon-ec2.html>

- Django Software Foundation**. Django Documentation** [última consulta: 08/02/2015] <https://docs.djangoproject.com/en/1.7/>

- Douglas Crockford. **JavaScript**: **The Good Parts**. O´Reilly Media, 2008

- González Duque, Raúl. **Python para todos**. Licencia Creative Commons, 2011.

- Google Inc. . **Angular Material** [última consulta: 10/02/2015] <https://material.angularjs.org/>

- Holovaty Adrian y Kaplan-Moss, Jacob. **The Django Book**. Apress, Diciembre 2007.

- W3schools. **Learn JavaScript and AJAX with W3Schools.** Wiley, Junio 2010.

Capítulo 7

7 Apéndice

7.1 Recurso Registro

7.1.1 Registro

Para realizar un registro se debe realizar una petición POST al servidor. Si la petición tiene éxito, se crea una instancia de Regist con ese correo (si no existía antes), un "username" que sale de recortar el email hasta el signo "@" y la contraseña. Finalmente, se envía un correo al usuario para que active su cuenta.

Request URL	http://127.0.0.1:8000/api/user/register/
Request Type	POST
Request Format	JSON
Request Authetication	None

Example Request

{
"email": EMAIL,
"password": PASSWORD,
"password_2": PASSWORD_2
}

Respuesta:

HTTP 201 CREATED

{
"email": EMAIL,
"activate": false
}

Errores:

Nombre	**Detalle**
"email"	"Este campo es obligatorio."
"email"	"Introduzca una dirección de correo electrónico válida".
"password"	"Este campo es obligatorio."
"password_2"	"Este campo es obligatorio."
"EmailActivado"	"Este email ya tiene una cuenta activa."
"EmailRegistrado"	"Este email ya está registrado. Revisa tu correo para activar tu cuenta."
"PasswordDiferente"	"Las passwords no coinciden."

"PasswordInvalida"	"Password muy corta. Mínimo: 6 caracteres."
"PasswordInvalida"	"Password muy larga. Máximo: 20 caracteres."
"PasswordInvalida"	"Formato de contraseña inválida (no permitido: espacios, /, \, %, @, =)."

7.2 Recurso Login/Logout

7.2.1 Login

El inicio de sesión se realiza mediante una petición POST enviando el email y la contraseña. Si los datos son correctos, el sistema nos devolverá un TOKEN, para ser utilizado cuando usemos ese tipo de autenticación, y nos creará una cookie llamada "sessionid" para la autenticación por sesión (la utilizada en este proyecto).

Request URL	http://127.0.0.1:8000/api/login/
Request Type	POST
Request Format	JSON
Request Authetication	None

Example Request

{
"email": EMAIL,
"password": PASSWORD
}

Respuesta:

HTTP 200 OK

{
"email": EMAIL,
"token": TOKEN
}

Errores:

Nombre	**Detalle**
"ErrorLogin"	"Password o usuario incorrectos."
"ErrorLogin"	"El usuario aún no ha sido activado".

7.2.2 Logout

Al hacer esta petición, el sistema eliminará las "cookies" creadas en el "login" y no podremos acceder al sistema.

Request URL	http://127.0.0.1:8000/logout/
Request Type	GET
Request Format	JSON
Request Authetication	None

7.3 Recurso Profile

En este recurso encontraremos todas las peticiones relacionadas con la gestión de un usuario.

7.3.1 Recuperar contraseña

Este recurso nos permite enviar un correo de restablecimiento de contraseña en el caso de que el usuario la haya olvidado y no pueda entrar al sistema.

Request URL	http://127.0.0.1:8000/api/forgot-password/
Request Type	POST
Request Format	JSON
Request Authetication	None

Example Request

```
{
  "email": EMAIL
}
```

Respuesta:

HTTP 200 OK

```
{
  "EmailEnviado": "Email enviado correctamente."
}
```

Errores:

Nombre	Detalle
“email”	"Este campo es obligatorio."
“ErrorUser”	"El usuario no está activado.”

7.3.2 Cambiar contraseña

Este recurso nos permite cambiar la contraseña de un usuario. Se realiza a través de una petición POST y, una vez la contraseña se ha cambiado correctamente, nos actualiza los credenciales dentro del sistema para que no tengamos que volver a hacer “login”.

Request URL	http://127.0.0.1:8000/api/change-password/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

{
“password”: PASSWORD,
“password_2”: PASSWORD_2
}

Respuesta**:**

HTTP 200 OK

{
“ContraseñaCambiada”: “Contraseña actualizada correctamente.”
}

Errores:

Nombre	Detalle
“password”	“Este campo es obligatorio.”
“password_2”	“Este campo es obligatorio.”
“PasswordDiferente”	“Las passwords no coinciden.”
"PasswordInvalida"	“Password muy corta. Mínimo: 6 caracteres.”
"PasswordInvalida"	“Password muy larga. Máximo: 20 caracteres.”
"PasswordInvalida"	“Formato de contraseña inválida (no permitido: espacios, /, \, %, @, =).”

7.3.3 Recuperar datos usuario

Para recuperar los datos de un usuario debemos hacer una petición GET, la que nos devolverá tanto el "nick" del usuario como los datos de usuario "email", "username", "first_name" y "last_name".

Request URL	http://127.0.0.1:8000/api/me/
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 200 OK

```
{
  "user": {
     "id": USER_ID,
     "first_name": "FIRST_NAME",
     "last_name": "LAST_NAME",
     "username": "USERNAME ",
     "email": "EMAIL"
   },
   "id": PROFILE_ID,
   "nick": "NICK"
}
```

7.3.4 Cambiar Nick

Este recurso nos permite actualizar el "nick" de un usuario. Por defecto, cuando un usuario activa su cuenta, el "nick" es el mismo que el "username".

Request URL	http://127.0.0.1:8000/api/nick/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

```
{
  "nick": NICK
}
```

Respuesta:

HTTP 200 OK

{
“NickCambiado”: “Nick actualizado correctamente.”
}

Errores:

Nombre	Detalle
“nick”	“Este campo es obligatorio.”
"NickInvalido"	“Nick muy corto. Mínimo: 4 caracteres.”
"NickInvalido "	“Nick muy largo. Máximo: 25 caracteres.”

7.4 Recurso Evento

En este recurso encontraremos todas las peticiones relacionadas con la gestión de los eventos.

7.4.1 Crear evento

Para crear un evento debemos realizar una petición POST con los campos “title”, “time_to_close” y “num_answers” mínimo. También podemos incluir “open” y “voters_public” si queremos; si no las introducimos, por defecto, serán “FALSE”.

El evento que creamos tiene que finalizar mínimo una hora después de la hora exacta en el momento en el que hacemos la petición, de lo contrario nos devolverá un error.

“num_answers” nos permite definir el número de opciones que puede votar un usuario diferenciando el tipo de la respuesta.

“open” nos permite definir si los invitados a un evento también pueden invitar a otros usuarios (“TRUE”) o si sólo el creador puede invitar (“FALSE”).

“voters_public” nos permite definir si se puede consultar quién ha votado cada opción (“TRUE”) o, por el contrario, si sólo se muestra el número de votos que tiene cada opción siendo éstos anónimos (“FALSE”).

Request URL	http://127.0.0.1:8000/api/event/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

```
{
  "title": "TITLE",
  "time_to_close":DATETIME (2014-11-11T21:02:54+01:00),
  "num_answers": NUM_ANSWER,
  "open": BOOLEAN,
  "voters_public": BOOLEAN
}
```

Respuesta:

HTTP 201 CREATED

```
{
  “is_owner”: true,
  “users”: [{ “id”: 1,
                “profile”:1,
                “event”:3}],
  "id": 3,
 "title": "Prueba API",
 "time_to_close": "2014-11-12T22:34:00+01:00",
 "num_answers": 1,
 "has_options": true,
 "has_dates": false,
 "open": false,
 "voters_public": false,
 "owner": 1
}
```

Errores:

Nombre	**Detalle**
“title”	“Este campo es obligatorio.”
“time_to_close”	“Este campo es obligatorio.”
“TimeError”	"El evento no puede finalizar en menos de 1 hora."
"TituloInvalido"	"Título muy corto. Mínimo 4 caracteres.”
"NumRespuestasInvalido"	"El número de respuestas permitido no puede ser 0."

7.4.2 Listar eventos

Este recurso devuelve todos los datos de los eventos en los que un usuario sea dueño o esté invitado. Los datos son devueltos por orden, desde los que antes finalizan a los que más tarde lo hacen, y nos informa de si somos los dueños de ese evento o no. También devuelve la lista de los usuarios que están participando en cada evento.
Por último, nos informa si somos los dueños del evento ("is_owner").

Request URL	http://127.0.0.1:8000/api/event/
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 200 OK


```
{
  "is_owner": True,
  "users": [{ "id": 1,
              "profile":1,
              "event":1}],
  "id": 1,
  "title": "Prueba API",
  "time_to_close": "2015-11-12T22:34:00",
  "num_answers": 1,
  "has_options": true,
  "has_dates": false,
  "open": false,
  "voters_public": false,
  "owner": 1
},
{
  "is_owner": False,
  "users": [{ "id": 2,
              "profile":1,
              "event":2}],
  "id": 2,
  "title": "Prueba API 2",
  "time_to_close": "2016-11-12T22:34:00",
  "num_answers": 1,
  "has_options": true,
  "has_dates": false,
  "open": false,
  "voters_public": false,
  "owner": 2
}
```

7.4.3 Detalle evento

Este recurso nos devuelve los detalles de un evento, entre los que se encuentra las respuestas que pertenecen a ese evento y los votos de cada respuesta.
Si el evento tiene las votaciones públicas, nos devuelve la lista que contiene quién ha votado cada opción. En el caso de que las votaciones no sean públicas, encontraremos un número entero en vez de una lista con el número total de votos que tiene esa opción.

Request URL	http://127.0.0.1:8000/api/event/ + ID_EVENTO
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 200 OK

```
{
  “is_owner”: True,
  “users”: [{ “id”: 1,
                “profile”:1,
                “event”:1}],
  “answer”:[{ “votes”: [{ “id”:5,
                            “voter”: USERNAME,
                            “vote”: 3
                          }],
                “id”:1,
                “answer”: “Respuesta 1”,
                “type”: “TX”,
                “event”: 1,
                “profile”: 2 }],
  "id": 1,
  "title": "Prueba API",
  "time_to_close": "2015-11-12T22:34:00",
  "num_answers": 1,
  "has_options": true,
  "has_dates": false,
  "open": false,
  "voters_public": false,
  "owner": 1
}
```

7.4.4 Edición evento

La edición se realizará mediante una petición PATCH, lo que permitirá actualizar únicamente los campos deseados.

Request URL	http://127.0.0.1:8000/api/event/ + ID_EVENTO
Request Type	PATCH
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

```
{
  FIELD_TO_UPDATE: FIELD_TO_UPDATE
}
```

Respuesta:

HTTP 200 OK

```
{
  “is_owner”: true,
  "id": 3,
 "title": "PRUEBA API",
 "time_to_close": "2014-11-12T22:34:00+01:00 ",
 "num_answers": 1,
 "has_options": true,
 "has_dates": false,
 "open": false,
 "voters_public": false,
 "owner": 1
}
```

Errores:

Nombre	Detalle
“TimeError”	"El evento no puede finalizar en menos de 1 hora."
"TituloInvalido"	"Título muy corto. Mínimo 4 caracteres.”
"NumRespuestasInvalido"	"El número de respuestas permitido no puede ser 0."

7.4.5 Eliminar evento

La eliminación de un evento produce un borrado en cascada en el que, además del evento, también se eliminarán los usuarios que están invitados al evento, las respuestas y los votos del mismo. Sólo puede eliminar un evento el creador del mismo.

Request URL	http://127.0.0.1:8000/api/event/ + ID_EVENTO
Request Type	DELETE
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 204 NO CONTENT

7.4.6 Crear evento junto con respuestas

Este recurso nos permite crear un evento con sus respuestas en una única petición.

Request URL	http://127.0.0.1:8000/api/evento/create_all/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

```
{
“event”:{
  "title": "TITLE",
  "time_to_close":DATETIME (2014-11-11T21:02:54+01:00),
  "num_answers": NUM_ANSWER,
  "open": BOOLEAN,
  "voters_public": BOOLEAN
},
“answers”:[
   {
    “answer”: “Respuesta 1”,
    “type”: “TX”
   },{
    “answer”: “Respuesta 1”,
    “type”: “TX”
   }
 ]
}
```

Respuesta:

HTTP 201 CREATED
“OK”

7.5 Recurso Gestión Usuarios en Eventos

En este recurso encontraremos todas las peticiones relacionadas con la gestión de los usuarios en los eventos.

7.5.1 Invitar evento

Mediante una petición POST podremos añadir participantes a un evento. Debemos especificar en el cuerpo de la petición el ID del evento y el ID del perfil.
Si la respuesta es correcta, se le enviará un correo al usuario invitado informándole de que ha sido invitado al evento.

Request URL	http://127.0.0.1:8000/api/share/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

{
“event”: EVENT_ID,
“profile”: PROFILE_ID
}

Respuesta:

HTTP 200 OK

{
"id": 1,
"profile": 1,
"event ": 1
}

Errores:

Nombre	**Detalle**
“event”	“Este campo es obligatorio.”
“profile”	“Este campo es obligatorio.”
“YaInvitado”	"Este usuario ya está invitado a este evento."
"ErrorInvitando"	"No puedes invitar usuarios a un evento que no es tuyo.”

"ErrorInvitando"	"No puedes invitar a un usuario que no sea amigo tuyo.”

7.5.2 Ver invitado eventos

Este recurso nos devuelve todos los eventos que te pertenecen o a los que estás invitado.

Request URL	http://127.0.0.1:8000/api/share/
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 200 OK

[{
“id”: 1,
“profile”:1,
“event”:1
},
{
“id”: 2,
“profile”:1,
“event”:2
}]

7.5.3 Eliminar invitación evento

Con una petición DELETE podemos revocar la participación de un usuario en un evento.

Sólo se permite el borrado de invitaciones sobre eventos de los que eres dueño.

Request URL	http://127.0.0.1:8000/api/share/ + ID_USEREVENT
Request Type	DELETE
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 204 NO CONTENT

7.6 Recurso Gestión Amigos

Este recurso nos permite realizar toda la gestión de amigos sobre un usuario.

7.6.1 Enviar solicitud de amistad

Request URL	http://127.0.0.1:8000/api/friend_request/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

{
“to_friend”: EMAIL
}

Respuesta:

HTTP 201 CREATED

{
"to_friend ": EMAIL
}

Errores:

Nombre	Detalle
“__all__”	"Friend request con éste To friend y From friend ya existe."
"ErrorPeticion"	"No te puedes enviar una solicitud de amistad a ti mismo.”
"Error petición"	“Ya sois amigos.”
“PeticionExistente”	“Ya has enviado una petición de amistad a este usuario.”
"PeticionExistente "	"Ese usuario ya te ha enviado una solicitud de amistad.”
“ErrorBuscando”	“Este usuario no existe.”

7.6.2 Ver todas las peticiones de amistad

Este recurso nos devolverá todas las peticiones de amistad enviadas por un usuario y las recibidas que no han sido aceptadas.

Request URL	http://127.0.0.1:8000/api/friend_request/
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 200 OK

```
[{
  "id": ID,
 "from_friend":  {
       “nick”: NICK,
       “email”: EMAIL
   },
 "to_friend ": {
       “nick”: NICK,
       “email”: EMAIL
   },
  “timestamp”: “2014-11-14T18:34:12Z”,
  “accepted”: false
},
{
  "id": ID,
 "from_friend": {
       “nick”: NICK,
       “email”: EMAIL
   },
 "to_friend ": {
       “nick”: NICK,
       “email”: EMAIL
   },
  “timestamp”: “2014-11-13T13:34:12Z”,
  “accepted”: false
}]
```

7.6.3 Aceptar petición de amistad

Para aceptar una petición de amistad debemos realizar una petición PATCH. Esta petición actualizará el campo "accepted" a "TRUE" y creará la relación de amistad en el sistema.

Request URL	http://127.0.0.1:8000/api/friend_request/ + ID_PETICION
Request Type	PATCH
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

{
"accepted": true
}

Respuesta:

HTTP 200 OK

{
"id": ID,
"from_friend": {
"nick": NICK,
"email": EMAIL
},
"to_friend ": {
"nick": NICK,
"email": EMAIL
},
"timestamp": "2014-11-14T18:34:12Z",
"accepted": true
}

Errores:

Nombre	**Detalle**
"ErrorPeticion"	"No se puede modificar la petición de amistad."
"ErrorPeticion"	"Solicitud de amistad ya aceptada."

Sólo se pueden aceptar peticiones de amistad en las que tú pertenezcas al campo "to_friend".

7.6.4 Eliminar petición de amistad

Para rechazar o deshacer una petición de amistad debemos hacer una petición DELETE. Únicamente se pueden eliminar peticiones de amistad en las que el usuario pertenezca al campo “from_friend” o “to_friend”.

Request URL	http://127.0.0.1:8000/api/friend_request/ + ID_PETICION
Request Type	DELETE
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 204 NO CONTENT

7.6.5 Obtener lista de amigos

Este “endpoint” nos permite obtener todos los amigos que tiene un usuario.

Request URL	http://127.0.0.1:8000/api/friends/
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 200 OK

```
[{
  "id": 1,
 “friend”: {
   “nick”: NICK,
   “id”: PROFILE_ID,
   “email”: EMAIL
  },
  “created”: DATE_TIME
}]
```

7.6.6 Borrar amigo

Para borrar un amigo realizamos una petición DELETE. Este borrado también eliminará del sistema la petición de amistad que estaba guardada.

Request URL	http://127.0.0.1:8000/api/friends/ + ID_AMIGO
Request Type	DELETE
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 204 NO CONTENT

Sólo se pueden eliminar amigos en las que el usuario pertenezca al campo "from_friend" o "to_friend".

7.7 Recurso Respuestas

En este recurso encontraremos todas las peticiones relacionadas con la gestión de las respuestas de un evento.

7.7.1 Crear respuesta

Para asociar una respuesta a un evento debemos realizar una petición POST.
Los campos a enviar en el cuerpo de la petición son "event" que es el ID del evento al que queremos asociar la respuesta, "answer" que es la respuesta en si.

Por ultimo, el campo "type" que sólo puede tener dos valores: "TX" si se trata de una respuesta de tipo texto, o "DT" si se trata de una respuesta de tipo fecha.

Request URL	http://127.0.0.1:8000/api/answer/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

{
“answer”: ANSWER,
“type”: TYPE,
“event”: EVENT_ID
}

Respuesta:

HTTP 201 CREATED

{
"id": 1,
"answer": “Respuesta 1”,
"type ": “TX”,
“event”: 1,
“profile”: 1
}

Errores:

Nombre	Detalle
“answer”	“Este campo es obligatorio.”
“type”	“Este campo es obligatorio.”
“event”	“Este campo es obligatorio.”
"OpcionError"	"No puedes añadir opciones a este evento. No eres el dueño.”
"OpcionError"	“No puedes añadir opciones a este evento. No estas invitado”.
"OpcionError "	"No puedes añadir opciones vacías.”
“type”	“Escoja una opción válida. TYPE no es una de las opciones disponibles.”
“answer”	“Asegúrese de que este valor tenga menos de 64 caracteres.”

7.7.2 Ver respuestas

Este “endpoint” nos permite devolver todas las respuestas filtradas por el evento al que pertenecen, el cual se pasa por parámetro. El usuario sólo puede ver las respuestas de los eventos a los que está invitado.

Request URL	http://127.0.0.1:8000/api/answer/?evento=ID_EVENTO
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 200 OK

```
[{
  "id": ID,
 "answer": “Respuesta 1”,
 "type ": “TX”,
  “event”: EVENT_ID,
  “profile”: PROFILE_ID
},
{
  "id": ID,
 "answer": “Respuesta 2”,
 "type ": “TX”,
  “event”: EVENT_ID,
  “profile”: PROFILE_ID
}]
```

7.7.3 Editar respuesta

Mediante una petición PATCH podemos editar el campo “answer” de una respuesta.
Sólo se pueden modificar las respuestas que hayas creado tú mismo.

Request URL	http://127.0.0.1:8000/api/answer/ + ID_RESPUESTA
Request Type	PATCH
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

```
{
  “answer”: “Prueba”
}
```

Respuesta:

HTTP 200 OK

```
{
  "id": 1,
 "answer": “Prueba”,
 "type ": “TX”,
  “event”: 1,
  “profile”: 1
}
```

Errores:

Nombre	Detalle
“answer”	“Este campo es obligatorio.”
"OpcionError"	"No puedes modificar las opciones de otros usuarios.”
"OpcionError"	“No puedes modificar el tipo al que pertenece la opción”.
"OpcionError "	“No puedes modificar el evento al que pertenece la opción”.

7.7.4 Eliminar respuesta

Para eliminar una respuesta de un evento debemos realizar una petición DELETE.

Si no eres dueño del evento sólo puedes eliminar respuestas creadas por ti mismo; si eres dueño del evento, puedes eliminar cualquier respuesta del mismo, tanto si ha sido creada por ti como si no.

Request URL	http://127.0.0.1:8000/api/answer/ + ID_RESPUESTA
Request Type	DELETE
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 204 NO CONTENT

7.8 Recurso Votos

En este recurso encontraremos todas las peticiones relacionadas con la gestión de los votos de un usuario.

7.8.1 Enviar/Actualizar votos

Para enviar un voto realizamos una petición POST con un único campo llamado "votes". Este campo es una lista con los ID de las respuestas a las que el usuario quiere votar. Este mismo método se utilizará también para actualizar las votaciones ya que, cuando se ejecuta, primero borra todos los votos antiguos del usuario que hace la petición.

Request URL	http://127.0.0.1:8000/api/vote/
Request Type	POST
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Example Request

{
"votes": [ANSWERID_1, ANSWERID_2]
}

Respuesta:

HTTP 201 CREATED

{
"OK"
}

Errores:

Nombre	**Detalle**
"vote"	"Este campo es obligatorio."
"ErrorVotacion"	"No puedes votar esta opción. No estas invitado al evento."
" ErrorVotacion "	"Este evento ya ha finalizado."
" ErrorVotacion "	"No puedes votar más opciones de este tipo."
" ErrorVotacion "	"Ya has votado esta opción".

7.8.2 Eliminar voto

Sólo puede eliminar un voto el dueño del mismo.

Request URL	http://127.0.0.1:8000/api/vote/ + ID_VOTO
Request Type	DELETE
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 204 NO CONTENT

7.8.3 Listar votos propios en un evento

Esta funcionalidad permite obtener los votos de un usuario en un determinado evento.

Request URL	http://127.0.0.1:8000/api/vote/?evento= + ID_VOTO
Request Type	GET
Request Format	JSON
Request Authetication	SessionAuthentication / TokenAuthentication

Respuesta:

HTTP 201 CREATED

```
[{
   "id": 1,
  "voter": “username”,
  "vote ": 1
},
{
   "id": 2,
  "voter": “username”,
  "vote ": 2
}]
```